Pen Edit Controls

Description

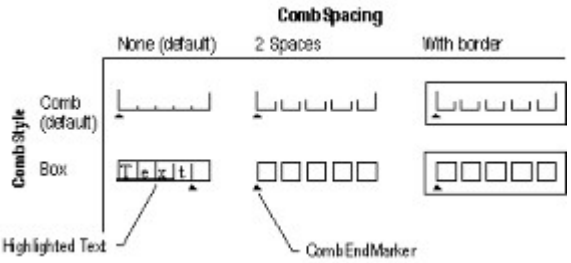
The pen edit custom controls allow for quick development of pen-aware applications under the Pen Services for Microsoft Windows 95 environment. The application designer may substitute these custom controls for the standard input controls available in Visual Basic. The pen edit custom controls are compatible with Visual Basic version 3.0 and Visual C++ version 1.5.

The handwriting edit control (HEdit) is a pen-enhanced version of the text box control. The handwriting edit control is shown here as it appears in the Toolbox.

The boxed edit control (BEdit) provides the application with comb or box guides that accept pen input. Each segment or box accepts only a single character of input. The boxed edit control is shown here as it appears in the Toolbox.

Note This control requires the Pen Services for Microsoft Windows 95 (Pen API version 2.0). It will not run under Microsoft Windows for Pen Computing, version 1.0.

These are typical BEdit controls:



File Name

PEN2CTL.VBX

Object Type

BEdit, HEdit

Remarks

The handwriting edit and boxed edit controls are similar to the standard Visual Basic text box, except that data can be entered into these controls using a pen in addition to the normal keyboard input method.

The handwriting edit control accepts free-form input from the user. This control supports most of the Visual Basic text box properties; however, it does not have dynamic data exchange (DDE) capabilities.

The boxed edit control expands upon the properties of the handwriting edit control and allows for additional manipulation of the writing area. This control accepts a single character of input in each box. This increases the accuracy of the recognition and in most cases is preferable to the handwriting edit control.

Distribution Note When you create and distribute applications that use the pen edit controls, you should install the file PEN2CTL.VBX in the customer's Microsoft Windows system subdirectory. The Visual Basic Setup Kit included with the Professional Edition in version 3.0 provides tools to help you

write setup programs that install your applications correctly.

Although many of the properties, events, and methods are similar, this release of the Visual Basic pen edit controls is not compatible with the previous release of the Pen Edit controls (in the Professional Edition of Visual Basic version 3.0). Applications written using the previous version will not automatically work with this version of the pen controls.

All of the properties, events, and methods for these controls are listed in the following table. Properties and events that apply *only* to these controls, or require special consideration when used with them, are marked with an asterisk (*). (Note that the list order is alphabetic from top to bottom, then left to right.) See the Visual Basic *Language Reference* or Help for documentation of the remaining properties, events, and methods.

Properties

About	*CombStyle(2)	*InflateBottom	SelStart
*AddWordList(2)	*DelayRecog	*InflateLeft	SelText
Alignment (1)	Enabled	*InflateRight	*SetBoxAlphabet(2)
*AltList(2)	*EraseInk	*InflateTop	*ShowAltList(2)
BackColor	FontBold	*InkColor	*SystemDict(2)
BorderStyle	FontItalic	*InkDataMode	TabIndex
*BoxAlphabetAlc(2)	FontName	*InkDataString	TabStop
*BoxCross(2)	FontSize	*InkWidth	Tag
*CellHeight(2)	FontStrikethru	*IntlPref	Text (3)
*CellWidth(2)	FontUnderline	Left	Top
*CharSet	ForeColor	MultiLine	Visible
*CharSetPriority	*GestureSet	Name	Width
*CombBaseLine(2)	Height	*NumBoxAlphabet(2)	*WordListCoercion(2)
*CombEndHeight(2)	HelpContextID	*OnTap	*WordListFile(2)
*CombHeight(2)	*hInk	Parent	*WordListStr(2)
*CombNumCols(2)	hWnd	*ReadWordList(2)	*WriteWordList(2)
*CombNumRows(2)	ImeMode	*ScrollBars	
*CombSpacing(2)	Index	SelLength	

(1) Applies only to HEdit control
(2) Applies only to BEdit control
(3) Text is the default value of the control.

Events

Change	GotFocus	KeyPress	*Update
DragDrop	LostFocus	KeyUp	
DragOver	KeyDown	*Result	

Methods

Move	**Refresh**	**SetFocus**	**ZOrder**

Functions

CPointerToVBType	**VBTypeToCPointer**

AddWordList Property

Applies To

BEdit controls.

Description

Add a word to the control's word list. This property is write-only and is available only at run time.

Visual Basic

[*form*.]*BEdit*.**AddWordList** = {**True** | **False**}

Visual C++

pBEdit->**SetNumProperty("AddWordList",** {**TRUE** | **FALSE**} **)**

Remarks

This property is used in conjunction with the WordListStr property. Setting the AddWordList property to **True** will add the word in the WordListStr property to the control's word list. Setting this property to **False** will have no effect. It is recommended that this property be called after the ReadWordList property so that all the words can be added to the same list. If ReadWordList is called after AddWordList, then a new word list is created, overwriting the existing word list.

Data Type

Integer (Boolean)

See Also

WordListStr property and ReadWordList property.

AltList Property

Applies To

BEdit controls.

Description

Sets or returns whether alternate lists are enabled or disabled.

Visual Basic

[*form*.]*BEdit*.**AltList** [**=** {**True** | **False**}]

Visual C++

pBEdit->**GetNumProperty("AltList")**
pBEdit->**SetNumProperty("AltList",** {**TRUE** | **FALSE**} **)**

Remarks

Setting the AltList property to **True** enables alternate character and word lists and setting it to **False** disable alternate lists. Similarly, a return value of **True** indicates that the alternate lists are enabled and a return value of **False** indicates that the alternate lists are disabled.

Data Type

Integer (Boolean)

BorderStyle Property

Applies To

BEdit and HEdit controls.

Description

Sets or returns the style of the control border. This property is read-only at run time (unlike the standard Visual Basic BorderStyle property).

Visual Basic

[*form*.]*Penctrl*.**BorderStyle**

Visual C++

pPenctrl->**GetNumProperty("BorderStyle")**

Remarks

The allowed BorderStyle settings for a BEdit control are the same as for the standard Visual Basic text and picture box controls. For the HEdit control, a new setting, underline, has been added. The underline setting may only be used on a single line HEdit control.

Setting	Description
0	None.
1	(Default) Fixed Single.
2	Underline (HEdit only).

Data Type

Integer (Enumerated)

BoxAlphabetAlc Property

Applies To

BEdit controls.

Description

Sets or returns the current alphabet code for the indexed box. This property is available only at runtime.

Visual Basic

[*form.*]*BEdit*. **BoxAlphabetAlc(i)**[= *setting%*]

Visual C++

pBEdit->**GetNumProperty("BoxAlphabetAlc(i)")**
pBEdit->**SetNumProperty("BoxAlphabetAlc(i)",** *setting* **)**

Remarks

This property is a one dimensional **Integer** array property, that stores the alphabet code (ALC) and is used to initialize the array. This property is used prior to the SetBoxAlphabet property to set the alphabet code for the range of boxes specified by the NumBoxAlphabet property. Note that it is an **Integer** array. For more information about box edit controls, refer to the *Programming Guide to Pen Services for Microsoft Windows 95*.

Data Type

Integer

See Also

SetBoxAlphabet property and NumBoxAlphabet property

BoxCross Property

Applies To

BEdit controls.

Description

Sets or returns whether box crosses in BEdit cells are enabled or disabled.

Visual Basic

[*form*.]*BEdit*.**BoxCross** [= {**True** | **False**}]

Visual C++

pBEdit->**GetNumProperty("BoxCross")**
pBEdit->**SetNumProperty("BoxCross**", {**TRUE** | **FALSE**} **)**

Remarks

Setting the BoxCross property to **True** displays box crosses in the BEdit cells and setting it to **False** removes them. Similarly, a return value of **True** indicates that the box crosses are displayed and a return value of **False** indicates that box crosses are not displayed. The default value is **False**.

Data Type

Integer (Boolean)

CellHeight, CellWidth Properties

Applies To

BEdit controls.

Description

Set or return the height and width of the encapsulating cell for a comb or box style BEdit control.

Visual Basic

[*form*.]*BEdit*.**CellHeight**[= *setting&*]
[*form*.]*BEdit*.**CellWidth**[= *setting&*]

Visual C++

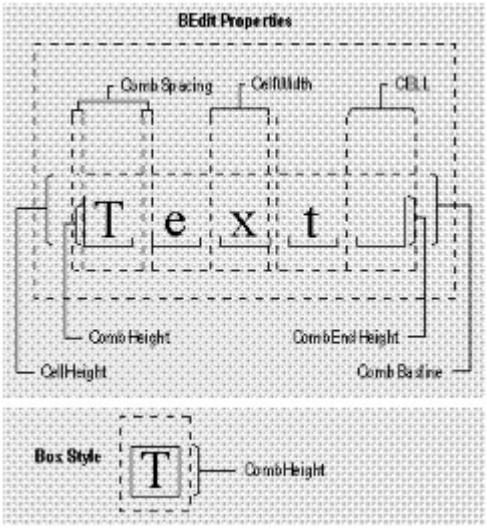
pBEdit->**GetNumProperty("CellHeight")**
pBEdit->**SetNumProperty("CellHeight",** *setting* **)**

Visual C++

pBEdit->**GetNumProperty("CellWidth")**
pBEdit->**SetNumProperty("CellWidth",** *setting* **)**

Remarks

CellHeight determines the maximum settings for the CombBaseLine, CombEndHeight, and CombHeight properties. The CellWidth value limits the CombSpacing values.



These properties use the ScaleMode property setting of the underlying form, frame, or picture box. For example, if a form's ScaleMode property is set to 3 (pixels), and a BEdit control is placed on that form, the CellHeight and CellWidth properties for that control are measured in pixels. In Visual C++, all dimemsions are in pixels.

Data Type

Long

See Also

CombBaseLine property, CombEndHeight property, CombHeight property and CombSpacing property.

CharSet Property

Applies To

BEdit and HEdit controls.

Description

Sets or returns the set of characters that will be recognized.

Visual Basic

[*form*.]*Penctrl*.**CharSet**[= *setting%*]

Visual C++

pPenctrl->**GetNumProperty("CharSet")**
pPenctrl->**SetNumProperty("CharSet",** *setting* **)**

Remarks

This property allows you to constrain the recognizer to specified sets of characters. For example, where it is known that a control will only accept numbers, the CharSet can be set to ALC_NUMERIC.

At design time, the CharSet property is set using the custom dialog. At run time, you must calculate the character set by summing the value of your choices (bitwise-inclusive-OR operation) according to the alphabet codes found in the PENAPI.TXT file (Visual Basic) or the PENWIN.H file (Visual C++).

The CharSet property settings are:

Setting	Description
ALC_DEFAULT	(Default) A recognizer-dependent set of characters. The default system-wide character set always includes alphanumeric, punctuation, white-space characters, and gestures.
ALC_LCALPHA	Lowercase alphabetic characters.
ALC_UCALPHA	Uppercase alphabetic characters.
ALC_NUMERIC	Numeric characters: 0 through 9.
ALC_PUNC	Punctuation: ! - ; ' " ? ( ) & : .
ALC_MATH	% ^ * ( ) - + = { } < >, / | .
ALC_MONETARY	Monetary symbols (for example: $, .)
ALC_OTHER	All symbols not included in the preceding sets, for example ¬ [ ] _ ~.
ALC_WHITE	Spaces between characters.
ALC_GESTURE	Gestures.

Data Type

Integer

See Also

CharSetPriority property.

CharSetPriority Property

Applies To

BEdit and HEdit controls.

Description

Sets or returns the current alphabet priority being used by the recognizer.

Visual Basic

[*form*.]*BEdit*.**CharSetPriority** [= *setting%*]

Visual C++

pBEdit->**GetNumProperty(** "**CharSetPriority**")
pBEdit->**SetNumProperty(** "**CharSetPriority**", *setting*)

Remarks

This property specifies the alphabet codes in the CharSet property that have the highest priority. For example, if the CharSet property specifies numeric and alpha characters, the CharSetProperty property could specify numeric characters so that in the event of a recogition confusion between an alpha and numeric, the numeric would take precedence.

For a listing of the alphabet codes available, use the CharSet property.

Data Type

Integer

See Also

CharSet property.

CombBaseLine Property

Applies To

BEdit controls.

Description

Sets or returns the distance from the top of the encapsulating cell to the base of the box or comb guide of a BEdit control.

Visual Basic

[*form*.]*BEdit*.**CombBaseLine**[= *setting&*]

Visual C++

pBEdit->**GetNumProperty("CombBaseLine")**
pBEdit->**SetNumProperty("CombBaseLine",** *setting* **)**

Remarks

CombBaseLine may range from 0 to the value of the CellHeight property. CombBaseLine also restricts the CombEndHeight and CombHeight properties. Refer to the diagram in the CellHeight, CellWidth property description.

In Visual Basic, this property uses the ScaleMode property setting of the underlying form, frame, or picture box. For example, if a form's ScaleMode property is set to 3 (pixels), and a BEdit control is placed on that form, the CombBaseLine property for that control is measured in pixels. In Visual C++, all dimensions are in pixels.

Data Type

Long

See Also

CellHeight property and CombHeight property.

CombEndHeight, CombHeight Properties

Applies To

BEdit controls.

Description

CombEndHeight sets or returns the height of the teeth in the comb in a comb-style BEdit control.

CombHeight sets or returns the height of the box in a box-style BEdit control.

Visual Basic

[*form.*]*BEdit.***CombEndHeight**[= *setting&*]
[*form.*]*BEdit.***CombHeight**[= *setting&*]

Visual C++

pBEdit->**GetNumProperty("CombEndHeight")**
pBEdit->**SetNumProperty("CombEndHeight",** *setting* **)**

pBEdit->**GetNumProperty("CombHeight")**
pBEdit->**SetNumProperty("CombHeight",** *setting* **)**

Remarks

The CombHeight property sets the height of the inner teeth of the comb. This setting is the height of the boxes when the BEdit control is in box style.

The CombEndHeight property sets the height of the beginning and end teeth of the comb. This property has no effect when the BEdit control is in box style.

Note The value of the CombHeight and CombEndHeight properties cannot exceed the CombBaseLine value.

In Visual Basic, these properties use the ScaleMode property setting of the underlying form, frame, or picture box. For example, if a form's ScaleMode property is set to 3 (pixels), and a BEdit control is placed on that form, the CombEndHeight and CombHeight properties for that control are measured in pixels. In Visual C++, all dimensions are in pixels.

See the diagram in the CellHeight, CellWidth property description.

Data Type

Long

See Also

CombBaseLine property and CellHeight property.

CombNumCols, CombNumRows Properties

Applies To

BEdit controls.

Description

Return the number of columns and rows displayed in a BEdit control. These properties not available at design time and are read-only at run time.

Visual Basic

[*form.*]*BEdit.***CombNumCols**
[*form.*]*BEdit.***CombNumRows**

pBEdit->**GetNumProperty("CombNumCols")**

pBEdit->**GetNumProperty("CombNumRows")**

Remarks

The CombNumCols property returns the number of columns in a BEdit control. The CombNumRows property returns the number of rows. These settings are dependent on the dimensions of the control and on the CellHeight and CellWidth properties. CombNumRows will always be 1 if the MultiLine property is set to **False**.

Data Type

Integer

CombSpacing Property

Applies To

BEdit controls.

Description

Sets or returns the distance between the side of the encapsulating cell walls and the side of the comb or box guide.

Visual Basic

[*form*.]*BEdit*.**CombSpacing**[**=** *setting*&]

Visual C++

pBEdit->**GetNumProperty("CombSpacing")**
pBEdit->**SetNumProperty("CombSpacing",** *setting* **)**

Remarks

The distance specified in the CombSpacing property is applied to both sides of the box and the comb guide area. CombSpacing cannot be greater than half the CellWidth.

This properties use the ScaleMode property setting of the underlying form, frame, or picture box. For example, if a form's ScaleMode property is set to 3 (pixels), and a BEdit control is placed on that form, the CombSpacing property for that control is measured in pixels.

See the diagram in the CellHeight, CellWidth property description.

Data Type

Long

CombStyle Property

Applies To

BEdit controls.

Description

Sets or returns the BEdit style type.

Visual Basic

[*form*.]*BEdit*.**CombStyle**[= *setting%*]

Visual C++

pBEdit->**GetNumProperty("CombStyle")**
pBEdit->**SetNumProperty("CombStyle",** *setting* **)**

Remarks

The CombStyle property settings are:

Setting	Description
0	(Default) Comb style.
1	Box style.

Data Type

Integer (Enumerated)

DelayRecog Property

Applies To

BEdit and HEdit controls.

Description

Determines whether the control recognizes writing immediately or leaves it as ink on the control for recognition at a later time.

Visual Basic

[*form*.]*Penctrl*.**DelayRecog** [= {**True | False**}]

Visual C++

pPenctrl->**GetNumProperty("DelayRecog")**
pPenctrl->**SetNumProperty("DelayRecog", {TRUE | FALSE})**

Remarks

The DelayRecog property settings are:

Setting	Description
False	(Default) Recognition is not delayed.
True	Recognition is delayed; writing remains as ink.

When DelayRecog is set to **True**, all writing remains as ink on the control until the property is set to **False**. When it is changed from **True** to **False**, the OnTap property is examined. If OnTap is **True**, recognition of the collected ink takes place when the user taps on the control with the pen; otherwise, recognition occurs immediately.

Any writing performed on the control while DelayRecog is set to **False** is recognized according to the user preferences set in the Control Panel.

Data Type

Integer (Boolean)

EraseInk Property

Applies To

BEdit and HEdit controls. This property is not available at design time.

Description

Setting the EraseInk property to **True** will erase any ink in a control if DelayRecog is **True**.

Visual Basic

[*form*.]*Penctrl*.**EraseInk**[= {**True | False**}]

Visual C++

pPenctrl->**GetNumProperty("EraseInk")**
pPenctrl->**SetNumProperty("EraseInk", {TRUE | FALSE})**

Remarks

If a control has DelayRecog set to **True**, the action of setting EraseInk to **True** causes any ink in the control to be erased. The property setting reverts back to **False** immediately after being set. It is used somewhat like a method rather than a property. Changing this property has no effect on a control if DelayRecog is set to **False**.

Data Type

Integer (Boolean)

See Also

DelayRecog property.

GestureSet Property

Applies To

BEdit and HEdit controls.

Description

Sets or returns the recognition of a specific gesture or a collection of gestures in the control.

Visual Basic

[form.]*BEdit*.**GestureSet**[= *setting%*]

Visual C++

pBEdit->**GetNumProperty("GestureSet")**
pBEdit->**SetNumProperty("GestureSet",** *setting* **)**

Remarks

At design time, the GestureSet property is set using the custom dialog. At run time, you must calculate the gesture set by summing the value of your choices (bitwise-inclusive-OR operation) according to the gesture codes found in the PENAPI.TXT file (Visual Basic) or the PENWIN.H file (Visual C++).

The following table lists the GestureSet property settings for the pen edit controls.

Global Constant	Value
GST_SEL	0x00000001L
GST_CLIP	0x00000002L
GST_WHITE	0x00000004L
GST_EDIT	0x00000010L
GST_SYS	0x00000017L
GST_CIRCLELO	0x00000100L
GST_CIRCLEUP	0x00000200L
GST_CIRCLE	0x00000300L
GST_ALL	0x00000317L

Data Type

Integer

hInk Property

Applies To

BEdit and HEdit controls.

Description

Returns a handle to an ink structure (**HPENDATA**) used by the Pen services. This property is not available at design time.

Visual Basic

[*form*.]*Penctrl*.**hInk**[= setting%]

Visual C++

pPenctrl->**GetNumProperty("hInk")**
pPenctrl->**SetNumProperty("hInk", *setting*)**

Remarks

The handle is provided by the operating environment and can be used to call Pen API functions that require a handle to a pen data structure. For more information on handles to pen data, refer to the *Programming Guide to Pen Services for Microsoft Windows 95*.

When the DelayRecog property is set to **FALSE** on a control, the hInk property is always 0.

Data Type

Integer

InflateBottom, InflateLeft, InflateRight, InflateTop Properties

Applies To

BEdit and HEdit controls.

Description

Set or return the area around a control's boundary onto which ink will be allowed.

Visual Basic

[form.]*Penctrl*.**InflateBottom**[= setting&]
[form.]*Penctrl*.**InflateLeft**[= setting&]
[form.]*Penctrl*.**InflateRight**[= setting&]
[form.]*Penctrl*.**InflateTop**[= setting&]

Visual C++

pPenctrl->**GetNumProperty("InflateBottom")**
pPenctrl->**SetNumProperty("InflateBottom",** *setting* **)**

pPenctrl->**GetNumProperty("InflateLeft")**
pPenctrl->**SetNumProperty("InflateLeft",** *setting* **)**

pPenctrl->**GetNumProperty("InflateRight")**
pPenctrl->**SetNumProperty("InflateRight",** *setting* **)**

pPenctrl->**GetNumProperty("InflateTop")**
pPenctrl->**SetNumProperty("InflateTop",** *setting* **)**

Remarks

These properties define a boundary around the control onto which ink can be placed after initially starting to ink within the control. Each of these properties must have a value greater than or equal to zero.

The settings are only applicable if the control is not in delayed recognition mode (that is, the DelayRecog property is **False**). In delayed-recognition mode, ink cannot be drawn outside a control's boundary.

Data Type

Long

InkColor Property

Applies To

BEdit, HEdit, and IEdit controls.

Description

Sets or returns the ink color.

Visual Basic

[*form*.]*Penctrl*.**InkColor**[= *color&*]

Visual C++

pPenctrl->**GetNumProperty("InkColor")**
pPenctrl->**SetNumProperty("InkColor",** *color* **)**

Remarks

The InkColor property settings are:

Setting	Description
&H8000000F&	(Default) Pen ink color as set in the Control Panel for Microsoft Windows for Pen Computing.
(color)	In Visual Basic, color specified by using the RGB scheme or the **QBColor** function in code. In Visual C++, use the MAKESYSCOLOR macro defined in the AFXEXT.H file to create system color constants. Standard RGB colors cab be used in Visual C++ as well.

The InkColor property, when read, only returns the value set by the InkColor property, not any value that might have been changed by the user in the Properties dialog or Default Properties dialog.

Data Type

Long

InkDataMode Property

Applies To

BEdit, HEdit, and IEdit controls.

Description

Sets or returns the mode that controls how the InkDataString data is used.

Visual Basic

[*form*.]*Penctrl*.**InkDataMode**[= *setting%*]

Visual C++

pPenctrl->**GetNumProperty("InkDataMode")**
pPenctrl->**SetNumProperty("InkDataMode",** *setting* **)**

Remarks

The InkDataMode property settings are:

Setting	Description
0	(Default) Replaces any ink that may be in the control when data is assigned to the InkDataString property.
1	Merges the currently displayed ink when ink data is assigned to the InkDataString property.

Data Type

Integer (Enumerated)

See Also

InkDataString property.

InkDataString Property

Applies To

BEdit, HEdit, and VIEdit controls.

Description

Sets or returns a string containing the compressed ink data associated with the control. This property is only available at run time.

Visual Basic

[*form*.]*Penctrl*.**InkDataString**[= *inkdatastring$*]

Visual C++

pPenctrl->**GetStrProperty("InkDataString")**
pPenctrl->**SetStrProperty("InkDataString",** *setting* **)**

Remarks

When a string is assigned to a control with DelayRecog set to **True**, the InkDataMode is checked to determine whether the new ink data replaces the existing ink, or if the new ink is merged with the existing ink.

If the control has DelayRecog set to **False** and ink data is assigned to the control's InkDataString property, then the ink is immediately recognized as if it were written on the control. If InkDataMode is set to '0 - Replace' when this assignment is done, then any existing text in the control is replaced by the result from recognizing the ink. Otherwise, the new recognition result is appended to the text in the control.

When DelayRecog is set to **False** on a control, the InkDataString property is always a null string ("").

Assigning invalid or uncompressed ink data to the InkDataString property generates a run-time error regardless of the setting DelayRecog.

Data Type

String (HLSTR)

See Also

InkDataMode property and DelayRecog property.

InkWidth Property

Applies To

BEdit and HEdit controls.

Description

Sets or returns the width of the ink that is drawn.

Visual Basic

[form.]Penctrl.**InkWidth**[= setting%]

Visual C++

pPenctrl->**GetNumProperty("InkWidth")**
pPenctrl->**SetNumProperty("InkWidth",** *setting* **)**

Remarks

The InkWidth setting defaults to -1, which tells the control to use the system default ink width (as set using the Control Panel). The valid range for the InkWidth property is 0 to 15 pixels. No ink is displayed when InkWidth is set to 0.

The InkWidth property, when read, only returns the value set by the InkWidth property, not any value that might have been changed by the user in the Properties dialog or Default Properties dialog.

Data Type

Integer

IntlPref Property

Applies To

BEdit and HEdit controls.

Description

Determines whether all ANSI characters are recognized or not.

Visual Basic

[*form*.]*Penctrl*.**IntlPref** [= {**True** | **False**}]

Visual C++

pPenctrl->**GetNumProperty("IntlPref")**
pPenctrl->**SetNumProperty("IntlPref",** {**TRUE** | **FALSE**} **)**

Remarks

Setting the *IntlPref* property to **True** allows the recognition of all ANSI characters. Setting it to **False** disables recognition of international characters.

Data Type

Integer (Boolean)

NumBoxAlphabet Property

Applies To

BEdit controls.

Description

Sets or returns the range of boxes to be used for recognition.

Visual Basic

[*form*.]*BEdit*.**NumBoxAlphabet**[= *setting%*]

Visual C++

pBEdit->**GetNumProperty("NumBoxAlphabet")**
pBEdit->**SetNumProperty("NumBoxAlphabet",** *setting* **)**

Remarks

The default value is 0. This property is used in conjunction with the BoxAlphabetAlc and the SetBoxAlphabet properties to set the alphabet code (ALC_ values) for the range of boxes (starting from the first box) in a BEdit control. The maximum allowable values is 31.

Data Type

Integer

See Also

BoxAlphabetAlc property and SetBoxAlphabet property.

OnTap Property

Applies To

BEdit and HEdit controls.

Description

Determines whether the control recognizes writing immediately upon changing the DelayRecog property from **True** to **False** or waits until the user taps the control.

Visual Basic

[*form.*]*Penctrl.***OnTap**[= {**True** | **False**}]

Visual C++

*pPenctrl->***GetNumProperty("OnTap")**
*pPenctrl->***SetNumProperty("OnTap",** *setting* **)**

Remarks

The OnTap property settings are:

Setting	Description
False	(Default) Recognition is not delayed after DelayRecog is set to **False.**
True	Recognition is delayed until the user taps the control (after DelayRecog is set to **False**).

When DelayRecog is set to **False**, this property has no effect on the control. Its state is examined only when the DelayRecog property is changed from **True** to **False**.

Data Type

Integer (Boolean)

See Also

DelayRecog property.

ReadWordList Property

Applies To

BEdit controls.

Description

Reads a word list from a file. This property is write-only and is available only at run time.

Visual Basic

[*form*.]*BEdit*.**ReadWordList** = {**True** | **False**}

Visual C++

pBEdit->**SetNumProperty("ReadWordList**", {**TRUE** | **FALSE**} **)**

Remarks

This property is used in conjunction with the WordListFile property. Setting the ReadWordList property to **True** creates a word list, if it does not already exist, and adds the words from the file specified by the WordListFile property. Setting this property to **False** has no effect.

It is recommended that this property be called before the AddWordList property so that all the word list can be read from a file and subsequently, words can be added to this word list. For more information on word lists refer to the Pen API documentation.

Data Type

Integer (Boolean)

See Also

WordListFile property and AddWordList property.

ScrollBars Property

Applies To

BEdit, HEdit, and IEdit controls.

Description

Determines if the control has horizontal or vertical scroll bars. This property is only available at design time.

Visual Basic

[*form*.]*Penctrl*.**ScrollBars**[= *setting%*]

Visual C++

pPenctrl->**GetNumProperty("ScrollBars")**
pPenctrl->**SetNumProperty("ScrollBars",** *setting* **)**

Remarks

The ScrollBars property settings are:

Setting	Description
0	(Default) None
1	Horizontal (HEdit only)
2	Vertical
3	Both (HEdit only)

The settings for the ScrollBars property in a BEdit or HEdit control operate in the same manner as the settings for a standard Visual Basic text box control.

Horizontal scroll bars are not allowed on BEdit controls.

For IEdit controls, the scroll bars enabled appear only when the ink drawn goes outside the boundaries of the control.

Data Type

Integer (Enumerated)

SetBoxAlphabet Property

Applies To

BEdit controls.

Description

Sets the alphabet code specified by the BoxAlphabetAlc property for the range of boxes specified by the NumBoxAlphabet property. This property is write-only and is available only at runtime.

Visual Basic

[*form.*]*BEdit*.**SetBoxAlphabet** = { **True** | **False** }

Visual C++

*pBEdit->***SetNumProperty("SetBoxAlphabet",** {**TRUE** | **FALSE**} **)**

Remarks

Setting the SetBoxAlphabet property to **True** will set the alphabet code for a range of boxes.

This property is used somewhat like a method rather than a property and is used in conjunction with the BoxAlphabetAlc and NumBoxAlphabet properties. To set the alphabet code (ALC) for a range of boxes in a BEdit control, set the BoxAlphabetAlc property to the alphabet code or codes desired, set the NumBoxAlphabet property to the range of boxes to which the alphabet code restriction will apply, and then set the SetBoxAlphabet property to **True** to set the alphabet code to those boxes.

Data Type

Integer (Boolean)

See Also

BoxAlphabetAlc property and NumBoxAlphabet property.

ShowAltList Property

Applies To

BEdit controls.

Description

Shows or hides the alternate character or word list. This property is not available at design time and is write-only at runtime.

Visual Basic

[*form.*]*BEdit*.**ShowAltList** = {**True** | **False**}

Visual C++

*pBEdit->***SetNumProperty("ShowAltList", {TRUE** | **FALSE**} **)**

Remarks

Setting the ShowAltList property to **True** displays the alternate character or word list and setting it to **False** will hide it.

Data Type

Integer (Boolean)

SystemDict Property

Applies To

BEdit controls.

Description

Sets or returns whether the default system dictionary is enabled or disabled.

Visual Basic

[*form*.]*BEdit*.**SystemDict** [= {**True** | **False**}]

Visual C++

pBEdit->**GetNumProperty("SystemDict")**
pBEdit->**SetNumProperty("SystemDict",** {**TRUE** | **FALSE**} **)**

Remarks

Setting the SystemDict property to **True** will enable the system dictionary that the system default recognizer will use and setting it to **False** will disable it. It returns **True** if the system dictionary is enabled and **False** otherwise.

This property is useful for disabling the system dictionary for a control when a word list is being used so that the recognizer uses only words from the word list.

Data Type

Integer (Boolean)

See Also

ReadWordList property, WordListFile property, and WriteWordList property.

WordListCoercion Property

Applies To

BEdit controls.

Description

Sets and returns the word list coercion to be used for word lists.

Visual Basic

[*form*.]*BEdit*.**WordListCoercion** [= *setting%*]

Visual C++

pBEdit->**GetNumProperty("WordListCoercion")**
pBEdit->**SetNumProperty("WordListCoercion",** *setting* **)**

Remarks

The WordListCoercion property is used to determine to what degree the recognizer will use the word list when performing recognition.

The following table lists the WordListCoercion property settings:

Setting	Description
0 - None	None. The word list is not used.
1 - Advise	(Default) The word list is used in conjunction with the system dictionary. The weighting of the word list and system dictionary is recognizer-dependent.
2 - Force	The word list is used exclusively. The recognizer is forced to return a word from the word list.

Data Type

Integer (Enumerated)

WordListFile Property

Applies To

BEdit controls.

Description

Sets or returns the word list file name.

Visual Basic

[*form*.]*BEdit*.**WordListFile**[= *setting$*]

Visual C++

pBEdit->**GetStrProperty("WordListFile")**
pBEdit->**SetStrProperty("WordListFile",** *setting* **)**

Remarks

This property specifies the filename of the file to which a word list is written. It is used in conjunction with the ReadWordList and WriteWordList properties to read and write the word list respectively. A word list is an application-defined list of words that the recognizer uses in attempting to match handwriting input.

Data Type

String

See Also

ReadWordList property and WriteWordList property.

WordListStr Property

Applies To

BEdit controls.

Description

Contains the word to be added to the word list. Available at run time only.

Visual Basic

[*form*.]*BEdit*.**WordListStr** [= *setting$*]

Visual C++

pBEdit->**GetStrProperty("WordListStr")**
pBEdit->**SetStrProperty("WordListStr",** *setting* **)**

Remarks

The WordListStr property contains a word that is to be added to the word list currently in use by the control. If it does not exist, a new word list for the control is created and the word is added to it. It is used in conjunction with the AddWordList property.

Data Type

String

See Also

AddWordList property.

WriteWordList Property

Applies To

BEdit controls.

Description

Writes a word list to a file. This property is write-only and is available at run time only.

Visual Basic

[*form*.]*BEdit*.**WriteWordList** = {**True** | **False**}

Visual C++

pBEdit->**SetNumProperty("WriteWordList**", {**TRUE** | **FALSE**} **)**

Remarks

Setting the WriteWordList property to **True** will write the word list to the file specified by the WordListFile property. It will overwrite any existing file with the same name. Setting this property to **False** will have no effect. For more information on word lists refer to *Programming Guide to Pen Services for Microsoft Windows 95*.

Data Type

Integer (Boolean)

See Also

WordListFile property.

Result Event

Applies To

BEdit and HEdit controls.

Description

Occurs whenever the control receives recognition results from the recognizer.

Syntax

Sub *Penctrl*_**Result (** *cSymbols* **As Integer,** *lpSymbols* **As Long)**

Remarks

The Result event occurs whenever the recognizer returns any recognition results to the control. The *cySymbols* parameter contains the number of recognized characters. The *lpSymbols* parameter contains a C pointer to a Visual Basic string that contains the recognized characters. The conversion from C pointer to Visual Basic string must be done by the application.

Update Event

Applies To

BEdit and HEdit controls.

Description

Occurs whenever the data in a control is changed.

Syntax

Sub *Penctrl*_**Update**( )

Visual C++

Function Signature**:**
void *CMyDialog::OnEditUpdate* **(UINT, int, Cwnd*, LPVOID)**

Remarks

The Update event occurs before the control redraws the data. This differs from the Change event that redraws the data before the event. Update can be used to format the new data so that flashes do not appear. An Update event does not occur when a Visual Basic program changes the text in the control using the Text property of the control.

VBTypeToCPointer Function

Description

Copies bytes from a Visual Basic variable memory location to a system memory location.

Visual Basic Syntax

VBTypeToCPointer(*vbSrc* **As Any, ByVal** *lpDest*, **ByVal** *cNum* **By Integer)**

Remarks

Copies *cNum* number of bytes from a memory location pointed to by *lpSrc* and places them in the *vbDest* memory location.

See Also

CPointerToVBType function.

CPointerToVBType Function

Description

Copies bytes from a system memory location to a Visual Basic variable memory location.

Visual Basic Syntax

CPointerToVBType(ByVal *lpSrc* **As Long,** *vbDest* **As Any, ByVal** *cNum* **By Integer)**

Remarks

Copies *cNum* number of bytes from a memory location pointed to by *lpSrc* and places them in the *vbDest* memory location.

See Also

VBTypeToCPointer function.

Pen Edit Controls Error Messages

The following table lists the trappable errors for the pen edit controls.

Error number	Message explanation
32001	**PENERR_INKWIDTH** **InkWidth must be in range 0-15 or -1 for default.** This error is caused by setting the InkWidth property to an invalid value.
32002	**PENERR_INVALIDINKDATA** **Invalid InkDataString format.** This error can occur when trying to assign invalid or uncompressed ink data to the InkDataString property.
32003	**PENERR_INFLATE** **Inflate value has to be greater than or equal to 0.** This error occurs if either InflateTop, InflateLeft, InflateRight, or InflateBottom property is set to a value less than 0.
32004	**PENERR_NEGCELLWIDTH** **CellWidth has to be greater than 0.** This error occurs if the CellWidth property is set to a value less than zero.
32005	**PENERR_CELLWIDTH** **CellWidth has to be greater than or equal to (CombSpacing * 2).** This error occurs if the CellWidth property is

	set to a value that is less than two times the value of the CombSpacing property. Try setting CombSpacing to 0 before setting the CellWidth property.
32006	**PENERR_NEGCELLHEIGHT** **CellHeight has to be greater than 0.** This error occurs if the CellHeight property is set to a value less than zero.
32007	**PENERR_CELLHEIGHT** **CellHeight has to be greater than or equal to CombBaseLine.** This error occurs if the CellHeight property is set to a value less than the CombBaseLine property. Change the CombBaseLine property before changing the CellHeight property.
32008	**PENERR_COMBSPACING** **CombSpacing out of range (0 CellWidth / 2).** This error occurs if the CombSpacing property is set to a value that is either less than zero or more than half the cell width.
32009	**PENERR_COMBBASELINE** **CombBaseLine out of range (0 - CellHeight).** This error occurs if the CombBaseLine property is set to a value that is either less

	than 0 or greater than the CellHeight property.
32010	**PENERR_COMBENDCUSP** **CombBaseLine must be greater than or equal to CombHeight and CombEndHeight.**
32011	**PENERR_COMBHEIGHT** **CombHeight out of range (0 - CombBaseLine).** This error occurs if the CombHeight property is set to a value that is either less than 0 or greater than the value of the CombBaseLine property.
32012	**PENERR_COMBENDHEIGHT** **CombEndHeight out of range (0 - CombBaseLine).** This error occurs if the CombEndHeight property is set to a value less than 0 or greater than the value of the CombBaseLine property.
32013	**PENERR_ADDWORDLIST** **Unable to add word to word list.** This error occurs if there is an error in adding a word, specified by the WordListStr property, to the current word list.
32014	**PENERR_READWORDLIST** **Unable to read word list from the specified file.** This error occurs if

there is an error in reading the word list from the file, specified by the WordListFile property.

32015	**PENERR_WRITEWORDLIST** **Unable to write word list to the specified file.** This error occurs if there is an error in writing the current word list to the file, specified by the WordListFile property.
32016	**PENERR_SETBOXALPHABET** **Unable to set the alphabet codes for the range of boxes.** This error occurs if the BoxAlphabetAlc cannot be set, possibly due to a problem in the NumBoxAlphabet or BoxAlphabetAlc property values.

Pen Ink Edit Control

Description

The pen ink edit control is an enhanced picture box control that allows the user to draw, erase, moved, resize, and manipulate pen strokes called *ink* on the control. It also allows you set background pictures and grid lines.

Note This control requires Microsoft Pen Services for Windows 95 (Pen API 2.0).

The pen ink edit control is shown here as it appears as an icon in the Toolbox.

File Name

PEN2CTL.VBX

Object Type

IEdit

All of the properties, events, and methods for this control are listed in the following table. Properties and events that apply *only* to this control, or require special consideration when used with it, are marked with an asterisk(*). (Note that the list order is alphabetic from top to bottom, then left to right.) See the Visual Basic *Language Reference* or Help for documentation of the remaining properties, events, and methods.

Properties

About	*FormatColor	hWnd	Name
*AppData	*FormatItem	*Image	Parent
*AutoSize	*FormatWidth	ImeMode	*Picture (1)
BackColor	*GridHeight	Index	*Recog
BorderStyle	*GridOrgLeft	*InkColor	*SaveUpStrokes
*CountSelStrokes	*GridOrgTop	*InkDataMode	*ScrollBars
*CountStrokes	*GridColor	*InkDataString	*Security
DragHandle	*GridStyle	*InkingMode	*SelectAll
DragIcon	*GridLineWidth	*InkInput	*StrokeIndex
*Draw	*GridWidth	*InkPicture	*StrokeSel
Enabled	Height	*InkRectHeight	TabIndex
*EraseInk	HelpContextID	*InkRectLeft	TabStop
*EraserColor	*hInk	*InkRectTop	Tag
*EraserWidth	*hInkGet	*InkRectWidth	Top
*Format	*hInkSet	*InkWidth	Visible
*FormatAttrb	*hMenu	Left	Width

(1) Picture is the default value of the control.

Events

Change	*Gesture	*ModeChanged
DragDrop	GotFocus	Update
DragOver	LostFocus	

Methods

Drag **Move** **Refresh** **Zorder**

Functions

CPointerToVBType **VBTypeToCPointer**

AppData Property

Applies To

IEdit controls.

Description

Sets or returns the user defined data of the control.

Visual Basic

[*form*.]*IEdit*.**AppData**[= setting&]

Visual C++

pIEdit->**GetNumProperty("AppData")**
pIEdit->**SetNumProperty("AppData",** *setting* **)**

Remarks

The pen ink edit control does not use this data in any way. It is provided for the application developer's convenience.

Data Type

Long

AutoSize Property

Applies To

IEdit controls.

Description

Determines the appearance of the background image in the pen ink edit control.

Visual Basic

[*form*.]*IEdit*.**AutoSize**[= *setting*%]

Visual C++

pIEdit->**GetNumProperty("AutoSize")**
pIEdit->**SetNumProperty("AutoSize",** *setting* **)**

Remarks

The AutoSize property settings are as follows:

Setting	Description
0	(Default) No autosizing takes place and the image is displayed in the upper-left corner of the control.
1	Automatically stretches the image to the size of the control.
2	Automatically adjusts the size of the control to exactly fit the bitmap.
3	Tiles the background image on the control.

Data Type

Integer (Enumerated)

CountSelStrokes Property

Applies To

IEdit controls.

Description

Returns the number of selected strokes.

Visual Basic

[*form*.]*IEdit*.**CountSelStrokes**

Visual C++

pIEdit->**GetNumProperty("CountSelStrokes")**

Remarks

This property is not available at design time and read-only at run time.

Data Type

Integer

CountStrokes Property

Applies To

IEdit controls.

Description

Returns the number of strokes in the control.

Visual Basic

[*form*.]*IEdit*.**CountStrokes**

Visual C++

pIEdit->**GetNumProperty("CountStrokes")**

Remarks

This property is not available at design time and read-only at run time.

Data Type

Integer

Draw Property

Applies To

IEdit controls.

Description

Sets or returns the drawing options.

Visual Basic

[*form*.]*IEdit*.**Draw**[= setting%]

Visual C++

pIEdit->**GetNumProperty("Draw")**
pIEdit->**SetNumProperty**(**"Draw",** *setting* **)**

Remarks

The following table lists the Draw property settings for the pen ink edit control:

Setting	Description
0	None
1	(Default) Fast.

The Visual Basic interface does not provide access to animated drawing, as does the C interface.

Data Type

Integer (Enumerated)

EraseInk Property

Description

Setting the EraseInk property to **True** erases any ink in the control. This property is not available at design time.

Visual Basic

[*form*.]*IEdit*.**EraseInk**[= {**True** | **False**}]

Visual C++

*pIEdit->***GetNumProperty(** "**EraseInk**" **)**
*pIEdit->***SetNumProperty(** "**EraseInk**", {**TRUE** | **FALSE**} **)**

Remarks

The property setting reverts back to **False** immediately after being set to **True**. It is used somewhat like a method rather than a property. The default color of the *erase ink* that is displayed while erasing is the nearest solid color to the BackColor property of the control.

Data Type

Integer (Boolean)

EraserColor Property

Applies To

IEdit controls.

Description

Sets and returns the eraser color.

Visual Basic

[*form*.]*IEdit*.**EraserColor**[= *color&*]

Visual C++

pIEdit->**GetNumProperty("EraserColor")**
pIEdit->**SetNumProperty("EraserColor",** *color* **)**

Remarks

By default, the window background color is used as the eraser color and it is not recommended that it be set to a different color.

The EraserColor property, when read, only returns the value set by the EraserColor property, not any value that might have been changed by the user in the Properties dialog or Default Properties dialog.

Data Type

Long

EraserWidth Property

Applies To

IEdit controls.

Description

Sets or returns the current eraser width.

Visual Basic

[*form*.]*IEdit*.**EraserWidth**[= *setting%*]

Visual C++

pIEdit->**GetNumProperty("EraserWidth")**
pIEdit->**SetNumProperty("EraserWidth",** *setting* **)**

Remarks

Acceptable values for the EraserWidth property are from 0 to 15 pixels, or -1 for default. By default it is set to 5 pixels.

The EraserWidth property, when read, only returns the value set by the EraserWidth property, not any value that might have been changed by the user in the Properties dialog or Default Properties dialog.

Data Type

Integer

Format Property

Applies To

IEdit controls.

Description

Actively sets the format of the strokes in the control. This property is write-only and is available only at run time.

Visual Basic

[*form.*]*IEdit*.**Format**[= {**True** | **False**}]

Visual C++

*pIEdit->***SetNumProperty("Format",** {**TRUE** | **FALSE**} **)**

Remarks

Setting the Format property to **True** immediately sets the format of the strokes in the control to the values specified by the current value of the FormatAttrb property based on the current value of the FormatItem property, which is used to specify which stroke or set of strokes are formatted. Setting this property to **False** has no effect.

Data Type

Integer (Boolean)

See Also

FormatItem property, FormatAttrb property, StrokeIndex property, StrokeSel property, FormatColor property and FormatWidth property.

FormatAttrb Property

Applies To

IEdit controls.

Description

Specifies or returns the attributes to be formatted by the Format property. This property is not available at design time and is read-write at run time.

Visual Basic

[*form.*]*IEdit*.**FormatAttrb**[= *setting%*]

Visual C++

pIEdit->**GetNumProperty("FormatAttrb")**
pIEdit->**SetNumProperty("FormatAttrb",** *setting* **)**

Remarks

This property setting is used by the Format property in conjunction with the FormatItem property to set the format of strokes.

The FormatAttrb property specifies whether to set the stroke color, width, or both. The FormatItem property specifies whether to set the FormatAttrib value for all strokes, selected strokes, or the stroke specified by the StrokeIndex property. A group of strokes can be selected prior to setting the Format property by setting the StrokeIndex property to each stroke and setting the StrokeSel property.

The following table lists the FormatAttrb property settings for the pen ink edit control:

Setting	Description
0	Set stroke color
1	Set stoke width
2	(Default) Set both stroke color and width.

To set or query the color or width to which the stroke(s) will be set, use the FormatColor and FormatWidth properties, respectively.

Data Type

Integer (Enumerated)

See Also

Format property, FormatItem property, FormatColor property, FormatWidth property, StrokeIndex property, and StrokeSel property.

FormatColor Property

Applies To

IEdit controls.

Description

Specifies the color to be used by the Format Property.

Visual Basic

[*form*.]*IEdit*.**FormatColor**[= *setting&*]

Visual C++

pIEdit->**SetNumProperty("FormatColor",** *setting* **)**

Remarks

This property determines the color that will applied to stroke or set of strokes when the FormatAttrb property specifies a color attribute and the Format property is set to **True**. The set of strokes to which this applies is determined by the FormatItem property.

The FormalColor property can be set to any color by using the RGB scheme or the **QBColor** function in code, or an offset into the system palette (like the standard color properties).

The FormatColor property, when read, only returns the value set by the FormatColor property, not any value that might have been changed by the user in the Properties dialog or Default Properties dialog.

Data Type

Long

See Also

Format property, FormatItem property, FormatAttrb property, FormatWidth property, StrokeIndex property, StrokeSel property.

FormatItem Property

Applies To

IEdit controls.

Description

Specifies or returns the stroke or set of strokes to be formatted by the Format property.

Visual Basic

[form.]*IEdit*.**FormatItem**[= *setting*%]

Visual C++

pIEdit->**GetNumProperty("FormatItem")**
pIEdit->**SetNumProperty("FormatItem",** *setting* **)**

Remarks

This property is used to determine the stroke or set of strokes that will be formatted when the Format property is set to **True**. This property is read-write at run time and is not available at design-time.

The following table lists the FormatItem property settings for the pen ink edit control:

Setting	Description
0	Set attributes for all the strokes.
1	(Default) Set attributes for all the selected strokes.
2	Set attributes of the stroke indexed by the StrokeIndex property setting.

Data Type

Integer (Enumerated)

See Also

Format property, FormatAttrb property, StrokeIndex property, FormatColor property, FormatWidth property, and StrokeSel property.

FormatWidth Property

Applies To

IEdit controls.

Description

Specifies the width in pixels to be used by the Format property.

Visual Basic

[*form*.]*IEdit*.**FormatWidth**[= *setting%*]

Visual C++

pIEdit->**GetNumProperty("FormatWidth")**
pIEdit->**SetNumProperty("FormatWidth",** *setting* **)**

Remarks

This property determines the width that will be applied to stroke or set of strokes when the FormatAttrb property specifies a width attribute and the Format property is set to **True**. The set of strokes to which this applies is determined by the FormatItem property.

Acceptable values for the FormatWidth property are from 0 to 15 pixels, or -1 for default.

Data Type

Integer

See Also

Format Property, FormatItem property, FormatAttrb property, FormatColor property, StrokeIndex property, and StrokeSel property.

GridHeight Property

Applies To

IEdit controls.

Description

Sets or returns the height (y-axis spacing) of the grid lines in pixels.

Visual Basic

[*form*.]*IEdit*.**GridHeight** [= *setting%*]

Visual C++

pIEdit->**GetNumProperty("GridHeight")**
pIEdit->**SetNumProperty("GridHeight",** *setting* **)**

Remarks

A setting of zero specifies no horizontal grid lines.

Data Type

Integer

See Also

GridOrgLeft property, GridOrgTop property, and GridWidth property.

GridOrgLeft Property

Applies To

IEdit controls.

Description

Sets or returns the x-axis origin of the grid in pixels.

Visual Basic

[*form*.]*IEdit*.**GridOrgLeft** [= *setting%*]

Visual C++

pIEdit->**GetNumProperty("GridOrgLeft")**
pIEdit->**SetNumProperty("GridOrgLeft",** *setting* **)**

Remarks

The default property setting is zero.

Data Type

Integer

See Also

GridOrgTop property, GridHeight property, and GridWidth property.

GridOrgTop Property

Applies To

IEdit controls.

Description

Sets or returns the y-axis origin of the grid in pixels.

Visual Basic

[*form*.]*IEdit*.**GridOrgTop** [= *setting%*]

Visual C++

pIEdit->**GetNumProperty("GridOrgTop")**
pIEdit->**SetNumProperty("GridOrgTop",** *setting* **)**

Remarks

The default property setting is zero.

Data Type

Integer

See Also

GridOrgLeft property, GridHeight property, and GridWidth property.

GridColor Property

Applies To

IEdit controls.

Description

Sets or returns the color of all lines used in the grid.

Visual Basic

[*form*.]*IEdit*.**GridColor** [= *color&*]

Visual C++

pIEdit->**GetNumProperty("GridColor")**
pIEdit->**SetNumProperty("GridColor", *color*)**

Remarks

The GridColor property can be set to any color by using the RGB scheme or the **QBColor** function in code, or an offset into the system palette (like the standard color properties).

Data Type

Long

See Also

GridStyle property and GridLineWidth property.

GridStyle Property

Applies To

IEdit controls.

Description

Sets or returns the style of the lines used in the grid.

Visual Basic

[*form*.]*IEdit*.**GridStyle** [= *value*]

Visual C++

pIEdit->**GetNumProperty("GridStyle")**
pIEdit->**SetNumProperty("GridStyle",** *setting* **)**

Remarks

The GridStyle property supports the following values.

Value	Style
0	Solid.
1	Dash.
2	Dot.
3	Dash dot.
4	Dash dot dot.

All lines with width greater than 1 appear as solid.

Data Type

Integer

See Also

GridStyle property and GridLineWidth property.

GridLineWidth Property

Applies To

IEdit controls.

Description

Sets or returns the width of the lines used in the grid.

Visual Basic

[*form*.]*IEdit*.**GridLineWidth** [= *value*]

Visual C++

pIEdit->**GetNumProperty("GridLineWidth")**
pIEdit->**SetNumProperty("GridLineWidth",** *setting* **)**

Remarks

The GridLineWidth property accepts any positive integer value as the number of pixels used for the width of each grid line.

This property is available at design time and read-write at run time.

Data Type

Integer

See Also

GridStyle property and GridColor property.

GridWidth Property

Applies To

IEdit controls.

Description

Sets or returns the width (x-axis spacing) of the grid lines in pixels.

Visual Basic

[form.]*IEdit*.**GridWidth** [= *setting*%]

Visual C++

pIEdit->**GetNumProperty("GridWidth")**
pIEdit->**SetNumProperty("GridWidth",** *setting* **)**

Remarks

A setting of zero specifies no vertical grid lines.

Data Type

Integer

See Also

GridOrgLeft property, GridOrgTop property, and GridHeight property.

hInk Property

Applies To

IEdit controls.

Description

Returns a handle to a pen data object (**HPENDATA**) used by the pen services. This property is not available at design time.

Visual Basic

[*form*.]*IEdit*.**hInk**[= *setting%*]

Visual C++

pIEdit->**GetNumProperty("hInk")**
pIEdit->**SetNumProperty("hInk",** *setting* **)**

Remarks

Use this handle when you need to make calls to Pen API functions that require a handle to an **HPENDATA** object. For details on handles to pen data objects, refer to the *Programming Guide to Pen Services for Microsoft Windows 95* .

Data Type

Integer

hInkGet Property

Applies To

IEdit controls.

Description

Sets or returns the option that controls what data is returned in the pen data object returned by the hInk property.

Visual Basic

[form.]*IEdit*.**hInkGet** [= setting%]

Visual C++

pIEdit->**GetNumProperty("hInkGet")**
pIEdit->**SetNumProperty("hInkGet",** *setting* **)**

Remarks

Following are the available property settings for the hInkGet property:

Setting	Description
0	(Default) Retrieve all the ink in the control.
1	Retrieve the selected ink in the control.

Data Type

Integer (Enumerated)

hInkSet Property

Applies To

IEdit controls.

Description

Sets or returns the option that controls how ink is added to the control by the hInk property.

Visual Basic

[form.]*IEdit*.**hInkSet** [= *setting*%]

Visual C++

pIEdit->**GetNumProperty("hInkSet")**
pIEdit->**SetNumProperty("hInkSet",** *setting* **)**

Remarks

The following table lists the hInkSet property settings for the pen ink edit control:

Setting	Description
0	(Default) Replace the ink in the control.
1	Append the ink to the existing ink in the control.

Data Type

Integer (Enumerated)

hMenu Property

Applies To

IEdit controls.

Description

Returns a handle to the menu used by the pen ink edit control. This property is not available at design time and is read-only at run time.

Visual Basic

[form.]*IEdit*.**hMenu**

Visual C++

pIEdit->**GetNumProperty("hMenu")**

Remarks

An application can perform all standard operations to this menu using standard Windows 95 API. However, it still belongs to the pen ink edit control and must not be deleted.

Data Type

Integer

Image Property

Applies To

IEdit controls.

Description

Returns a handle to a bitmap containing the combined Picture and InkPicture bitmaps.

Visual Basic

[*form*.]*IEdit*.**Image**

Visual C++

pIEdit->**GetNumProperty("Image")**

Remarks

This property is not available at design time and is read-only at run time.

Data Type

Integer

See Also

Picture property and InkPicture property.

InkDataMode Property

Applies To

IEdit, HEdit, and BEdit controls.

Description

Sets or returns the mode that controls how the InkDataString data is used.

Visual Basic

[*form*.]*IEdit*.**InkDataMode**[= *setting%*]

Visual C++

pIEdit->**GetNumProperty("InkDataMode")**
pIEdit->**SetNumProperty("InkDataMode",** *setting* **)**

Remarks

The InkDataMode property settings are as follows:

Setting	Description
0	(Default) Replaces any ink that may be in the control when data is assigned to the InkDataString property.
1	Merges the currently displayed ink when ink data is assigned to the InkDataString property.

Data Type

Integer (Enumerated)

See Also

InkDataString property.

InkDataString Property

Applies To

IEdit, HEdit, and BEdit controls.

Description

Sets or returns a string containing the compressed ink data associated with the control.

Visual Basic

[*form*.]*IEdit*.**InkDataString**[= *inkdatastring$*]

Visual C++

pIEdit->**GetStrProperty("InkDataString")**
pIEdit->**SetStrProperty("InkDataString",** *setting* **)**

Remarks

When a string is assigned to a control, the InkDataMode is checked to determine whether the new ink data replaces the existing ink or is merged with the existing ink. Assigning invalid or uncompressed ink data to the InkDataString property generates a run-time error.

This property is only available at run time.

Data Type

String

See Also

InkDataMode property.

InkingMode Property

Applies To

IEdit controls.

Description

Sets or returns the currently selected inking mode of the pen ink edit control. This property is available only at run time.

Visual Basic

[*form*.]*IEdit*.**InkingMode**[= *setting%*]

Visual C++

pIEdit->**GetNumProperty("InkingMode")**
pIEdit->**SetNumProperty("InkingMode",** *setting* **)**

Remarks

The InkingMode property settings are:

Setting	Description
0 - Ready	Pen tip is set to draw ink, erase, or resize.
1 - Erase	Pen tip is set to erase ink.
2 - Lasso Selection	Pen tip is set to select an area by drawing around the area.

The erase color and erase width are set by the EraserColor and EraserWidth properties, respectively. When set in lasso selection mode, the control uses a solid black pen with an pen width of 1. Following a successful mode change, the ModeChanged event is triggered.

Data Type

Integer (Enumerated)

InkInput Property

Applies To

IEdit controls.

Description

Sets or returns the ink input options that control the inking in the pen ink edit control.

Visual Basic

[form.]IEdit.**InkInput** [= *setting*%]

Visual C++

pIEdit->**GetNumProperty("InkInput")**
pIEdit->**SetNumProperty("InkInput",** *setting* **)**

Remarks

At design time, the InkInput property is set using the Ink Input dialog, which allows you to select check boxes for Move, Resize, Crop or Discard. At run time, the property must be calculated by using the bitwise-OR operator to combine the desired values according to the values in the PENAPI.TXT (for Visual Basic) or PENWIN .H (for Visual C++) file.

The following table lists the InkInput property settings for the pen ink edit control:

Setting	Value	Description
IEI_MOVE	&H1	Move ink into control.
IEI_RESIZE	&H2	Resize ink to fit within control.
IEI_CROP	&H4	Discard ink outside of control.
IEI_DISCARD	&H8	Discard all ink if any outside control.

The default value of this property is (IEI_MOVE | IEI_RESIZE).

Data Type

Integer

InkPicture Property

Applies To

IEdit controls.

Description

Returns a handle to a bitmap containing an image of the ink in the color.

Visual Basic

[*form*.]*IEdit*.**InkPicture**

Visual C++

pIEdit->**GetNumProperty("InkPicture")**

Remarks

This property is not available at design time, and is read-only at run time.

Data Type

Integer

InkRectHeight Property

Applies To

IEdit controls.

Description

Returns the height in pixels of the bounding rectangle of the ink in the pen ink edit control.

Visual Basic

[*form*.]*IEdit*.**InkRectHeight**

Visual C++

pIEdit->**GetNumProperty("InkRectHeight")**

Remarks

This property is not available at design time and is read-only at run time.

Data Type

Integer

InkRectLeft Property

Applies To

IEdit controls.

Description

Returns the left x-axis position in pixels of the bounding rectangle of the ink in the pen ink edit control.

Visual Basic

[*form*.]*IEdit*.**InkRectLeft**

Visual C++

pIEdit->**GetNumProperty("InkRectLeft")**

Remarks

This property is not available at design time and is read-only at run time. When scrolled, the left x-axis value is offset by the amount scrolled.

Data Type

Integer

InkRectTop Property

Applies To

IEdit controls.

Description

Returns the top y-axis position in pixels of the bounding rectangle of the ink in the pen ink edit control.

Visual Basic

[*form*.]*IEdit*.**InkRectTop**

Visual C++

pIEdit->**GetNumProperty("InkRectTop")**

Remarks

This property is not available at design time and is read-only at run time. When the control is scrolled, the top y-axis position value is offset by the amount scrolled.

Data Type

Integer

InkRectWidth Property

Applies To

IEdit controls.

Description

Returns the width in pixels of the bounding rectangle of the ink in the pen ink edit control.

Visual Basic

[*form*.]*IEdit*.**InkRectWidth**

Visual C++

pIEdit->**GetNumProperty("InkRectWidth")**

Remarks

This property is not available at design time and is read-only at run time.

Data Type

Integer

InkWidth Property

Applies To

IEdit controls.

Description

Sets or returns the width of the ink.

Visual Basic

[*form*.]*IEdit*.**InkWidth**[= *setting%*]

Visual C++

pIEdit->**GetNumProperty("InkWidth")**
pIEdit->**SetNumProperty("InkWidth",** *setting* **)**

Remarks

The InkWidth property settings are:

Setting	Description
-1	(Default) Sets the ink width to the default system ink width (as defined in the Control Panel).
0	No ink is displayed.
1 to 15	The range of valid, visible ink widths in pixels.

The InkWidth property, when read, only returns the value set by the InkWidth property, not any value that might have been changed by the user in the Properties dialog or Default Properties dialog.

Data Type

Integer

Picture Property

Applies To

IEdit controls.

Description

Specifies the graphic to be displayed as the background image in the pen ink edit control. This property is write-only at design time.

Visual Basic

[*form*.]*IEdit*.**Picture**[= *picture%*]

Visual C++

pIEdit->**SetPictureProperty("Picture",** *picture* **)**
pIEdit->**SetPictureProperty("Picture",** *picture* **)**

Remarks

The Picture property settings are as follows:

Setting	Description
(none)	(Default) No image is used in the background.
(Bitmap)	Designates that a bitmap is displayed in the background.

In Visual Basic, you can load a graphic at design time from the Properties window. At run time, you can set this property by using the **LoadPicture** function on a bitmap or icon or you can use Clipboard methods such as **GetData**, **SetData**, and **GetFormat** with nontext Clipboard formats CF_BITMAP and CF_DIB, as described in the Visual Basic version 3.0 CONSTANT.TXT file.

This control can display bitmaps (.BMP) but not Windows metafiles (.WMF). At run time, you can set the Picture property to any other object's Picture, or Image property, or you can assign it the graphic returned by the **LoadPicture** function. You can only assign the Picture property directly.

Data Type

Integer

Recog Property

Applies To

IEdit controls.

Description

Sets or returns the recognition options that control recognition in the pen ink edit control.

Visual Basic

[form.]*IEdit*.**Recog** [= *setting*%]

Visual C++

pIEdit->**GetNumProperty("Recog")**
pIEdit->**SetNumProperty("Recog",** *setting* **)**

Remarks

The following table lists the Recog property settings for the pen ink edit control:

Setting	Description
0	No recognition
1	Gesture recognition
2	(Default) All recognition. This is the same as Gesture recognition for this version.

Data Type

Integer (Enumerated)

SaveUpStrokes Property

Applies To

IEdit controls.

Description

Starts or stops accumulation of upstrokes of pen data.

Determines whether upstrokes of pen data (which are not displayed) are saved by the control.

Visual Basic

[*form*.]*IEdit*.**SaveUpStrokes** [= {**True** | **False**}]

Visual C++

pIEdit->**GetNumProperty("SaveUpStrokes")**
pIEdit->**SetNumProperty("SaveUpStrokes", {TRUE** | **FALSE**} **)**

Remarks

Setting the SaveUpStrokes property to **True** begins the accumulation or upstokes (which are not displayed) in the control. Setting SaveUpStrokes to **False**, ends the accumlation. This affects all ink drawn after setting the control. Existing ink is unaffected.

Data Type

Integer (Boolean)

Security Property

Applies To

IEdit controls.

Description

Sets or returns the security options set for the pen ink edit control.

Visual Basic

[form.]IEdit.**Security** [= *setting*%]

Visual C++

pIEdit->**GetNumProperty("Security")**
pIEdit->**SetNumProperty("Security",** *setting* **)**

Remarks

At design time, the Security property is set using the custom dialog. At run time, the property must be calculated by using the bitwise-OR operator to combine the desired values according to the codes in the PENAPI.TXT (Visual Basic) or PENWIN.H (Visual C++) file. By default, the control is completely unsecured.

The Security property settings for the pen ink edit control are as follows:

Setting	Value	Description
IESEC_NOCOPY	&H1	Disable copying of ink.
IESEC_NOCUT	&H2	Disable cutting of ink.
IESEC_NOPASTE	&H4	Disable pasting of ink.
IESEC_NOUNDO	&H8	Disable undo action.
IESEC_NOINK	&H10	Disable inking.
IESEC_NOERASE	&H20	Disable erasing.
IESEC_NOGET	&H40	Disable reading hInk property.
IESEC_NOSET	&H80	Disable writing hInk property.

Data Type

Integer

SelectAll Property

Applies To

IEdit controls.

Description

Setting the SelectAll property to **True** will select all ink in the control and setting it to **False** will deselect all ink in the control. This property is not available at design time and is write only at run time.

Visual Basic

[*form.*]*IEdit*.**SelectAll** ={**True** | **False**}

Visual C++

*pIEdit->***SetNumProperty("SelectAll", {TRUE** | **FALSE**} **)**

Data Type

Integer (Boolean)

StrokeIndex Property

Applies To

IEdit controls.

Description

Sets or returns the zero-based stroke index to be formatted or selected. This property is not available at design time.

Visual Basic

[*form.*]*IEdit*.**StrokeIndex** [= *setting%*]

Visual C++

pIEdit->**GetNumProperty("StrokeIndex")**
pIEdit->**SetNumProperty("StrokeIndex", *setting*)**

Remarks

This property is used by the Format property only if the FormatItem property is set to 2. It is also used by the StrokeSel property to set or reset the selection status of a stroke.

Data Type

Integer

See Also

StrokeSel property, Format property, and FormatItem property.

StrokeSel Property

Applies To

IEdit controls.

Description

Sets or returns the selection status of the stroke specified by the StrokeIndex property. This property is not available at design time.

Visual Basic

[*form.*]*IEdit*.**StrokeSel** [= {**True** | **False**}]

Visual C++

*pIEdit->***GetNumProperty("StrokeSel")**

*pIEdit->***SetNumProperty("StrokeSel",** {**TRUE** | **FALSE**} **)**

Remarks

Setting the StrokeSel property to **True** will select the stroke specified by the StrokeIndex property and setting it to **False** will deselect it. Similarly, a return value of **True** indicates that the specified stroke is selected; **False** indicates the specified stroke is deselected.

Data Type

Integer (Boolean)

See Also

StrokeIndex property

Ink Edit Events

Change Event

Description

Occurs whenever the data in the control is changed.

Visual Basic

Sub *IEdit*_**Change**()

Visual C++

Function Signature**:**
void *CMyDialog::OnIEditChange* **(** *UINT*, *int*, *Cwnd**, *LPVOID* **)**

Remarks

The Change event occurs after the control redraws the data in the control.

Gesture Event

Description

Occurs when new ink entered into the control is recognized as a gesture.

Visual Basic

Sub *IEdit*_**Gesture(** *HrcResult* **As Long)**

Visual C++

FunctionSignature**:**
void *CMyDialog::OnIEditGesture* **(** *UINT*, *int*, *Cwnd**, *LPVOID lpParams* **)**

Parameter Usage:
AFX_NUM_EVENTPARAM (*LONG*, *lpParams* **)**

Remarks

The Gesture event does not get fired if recognition is turned off, which can be done by setting the Recog property to 0. This event does not return the gesture directly. The *HrcResult* parameter is a handle to a handwriting recognition context result object (**HRCRESULT**) from which the gesture can be retrieved. This handle is valid only during the processing of this event. The handle that is passed in belongs to the IEdit control and must not be modified in any way.

For more information on using HRCRESULT objects, see the *Programming Guide to Pen Services for Microsoft Windows 95*.

ModeChanged Event

Description

Occurs whenever the pen mode is changed.

Visual Basic

Sub *IEdit*_**ModeChanged**()

Visual C++

Function Signature**:**
void *CMyDialog::OnIEditModeChanged* **(UINT, int, Cwnd*, LPVOID)**

Remarks

The ModeChanged event occurs after the pen mode is changed. Ink modes can be changed programmatically using the InkingMode property, or by user's actions.

Update Event

Description

Occurs whenever the data in the control is changed.

Visual Basic

Sub *PenCtrl*_**Update**()

Visual C++

Function Signature**:**
void *CMyDialog::OnPenCtrlUpdate* **(UINT, int, Cwnd*, LPVOID)**

Remarks

The Update event occurs before the control redraws the data. This differs from the Change event, which redraws the data before the event. You can use the Update event to format the new data so that flashes do not appear.

IEdit Error Messages

The following table lists the trappable run-time errors for the pen ink edit controls.

Error number	Message explanation
32001	**PENERR_INKWIDTH** **InkWidth must be in range 0 – 15 or - 1 for default.** This error is caused by setting the InkWidth property to an invalid value.
32002	**PENERR_INVALIDINKDATA** **Invalid InkDataString format.** This error can occur when trying to assign invaoid or uncompressed ink data to the InkDataString property.
32017	**PENERR_INVALIDPICTURE** **Picture format not supported.** This error is caused by setting the Picture property to an invalid value. The control can display only bitmap (.BMP) format files.
32018	**PENERR_DISPLAYFAILED** **Unable to display bitmap.** The control is unable to display the bitmap. This error can be caused by low memory.
32019	**PENERR_SECURITY** **Cannot get/set property with current Security setting.** This error is caused by either querying or setting the hInk property with the

	IESEC_NOTGET or IESEC_NOSET bits set in the Security property.
32020	**PENERR_SETINK** **Unable to set ink.** This error is generated if the ink cannot be set into the control, which is normally due to a low memory condition.
32021	**PENERR_STROKEINDEX** **StrokeIndex must be less than the number of strokes or 0.** This error is caused by specifiying an invalid stroke index, mainly by specifying an index greater than the number of strokes in the pen data.
32022	**PENERR_SETMODE** **Cannot set this mode with no pendata in the control.** This error is generated when it is not possible to set the pen in the specified mode, for example, when trying to set Erase mode or Lasso mode with no ink in the control.
32023	**PEN_ERR_GRIDPEN** **Unable to set grid pen.** This error is generated when attempting to set invalid grid properties.
32024	**PENERR_INK** **Invalid or null pen data object. Unable to get or set property.** This error is generated when attempting to access properities of a pen data object that is invalid.